Com Sci 31, Winter 2019

Project 1 – Report File

Name – Suyash Kumar

UID – 605108040

# Input producing anomalous results in original.cpp

* Using input sets which contain negative values, such as -10 for Democrats, 10 for Republicans, and 2 for Independents produces negative percentages, and possibly percentages greater than 100.0 (in this case Democrats: 500.0%, Republicans: -500.0%, and Independents: 100.0%).
* Putting in 0 for all input values produces -nan% for all categories in g31 and nan% for all categories in XCode. In this case, the statement of the *else* construct is returned (“Looks like control goes to the Democrats”).
* Putting in equal values for each category, say 12, returns the statement of the *else* construct.
* The following interesting observations are made for out-of-range integer inputs -

1. In XCode
2. on feeding a whole number like 12345678901234567890, which exceeds the size limit of *int* datatype, for Democrats, the console skips demanding input for the next two categories and produces the following percentages – Democrats: 114.5%, Republicans: 0.0%, Independents: -14.5%. In fact, any out of range input in the first category of Democrats (like 892489248294248924989) makes the program skip taking in further inputs and produces the same result as above.
3. Putting in a reasonable integer for Democrats, say 43, and an out of range integer for Republicans, like 890982109821049821, makes the program skip demanding input for Independents, and produces a 0.0% for Democrats, 114.5% for Republicans, and -14.5% for Independents. Using a similar input set, like 18 for Democrats, and 21301924809812409 for Republicans, leads to the same set of events as the previous input set triggered.
4. Finally, input sets containing reasonable integers for both Democrats and Republicans but out of range values for Independents (like Democrats: 18, Republicans: 24, Independents: 32520395823098, or Democrats: 96, Republicans: 74, Independents: 824768317641837) produces -0.0% for both Democrats and Republicans and -100.0% for Independents.
5. In g31 however, for out-of-size integer inputs, the other inputs are skipped in a similar fashion as above, however the outputs are -100.0%, -0.0%, and -0.0%, with -100.0% being the output percentage for the out-of-range integer input.

# Logical errors introduced in logic\_error.cpp

* The formula for pctRep divides by (total+7) instead of total. (Line 22)
* The formula for pctDem multiplies by 50.0 instead of 100.0. (Line 23)
* The formula for pctInd multiplies by (IndependentSeats+19) instead of IndependentSeats. (Line 24)
* The if construct uses the condition pctRep<pctDem instead of pctRep>pctDem (Line 33)

# Compilation errors introduced in compile\_error.cpp

* Semi-colon has been left out in line 15. The compiler of XCode IDE reports “Expected ‘;’ after the expression”. The g31 compiler however reports expected “'**;**' before '**cin**'”.
* Only < is used instead of << in line 28. That is, whereas the code should have been ‘cout<<endl<<endl;’ it is ‘cout<endl<<endl;’. The XCode compiler reports “Reference to the overloaded function could not be resolved; did you mean to call it?”. The g31 compiler reports, however, “invalid operands of types '**<unresolved overloaded function type>**' and '**<unresolved overloaded function type>**' to binary '**operator<<**'”.
* end1 has been used instead of endl in line 31. The compiler reports “Use of undeclared identifier ‘end1’;”. However, the g31 compiler reports, “'**end1**' was not declared in this scope”.